AMENDMENTS TO THE SPECIFICATION

Please amend the Specification as shown below. Applicant respectfully submits that the proposed amendments are to correct various informalities in the Specification, and that no new matter is being added.

Please replace paragraphs [0005] through [0010] with the following amended paragraphs:

[0005] FEDERATED MANAGEMENT OF CONTENT REPOSITORIES, U.S. Application No. [[_____]] 10/618,513, Inventors: James Owen, et al., filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1360US1)

[0006] VIRTUAL REPOSITORY CONTENT MODEL, U.S. Application No. [[_____]] 10/618,519, Inventors: James Owen, et al., filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1361US0)

[0007] VIRTUAL REPOSITORY COMPLEX CONTENT MODEL, U.S. Application No. [[_____]] 10/618,380, Inventors: James Owen, et al., filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1364US0)

[0008] VIRTUAL CONTENT REPOSITORY APPLICATION PROGRAM INTERFACE, U.S. Application No. [[_____]] 10/618,494, Inventors: James Owen, et al., filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1370US0)

[0009] SYSTEM AND METHOD FOR SEARCHING A VIRTUAL REPOSITORY CONTENT, U.S. Application No. [[_____]] 10/619,165, Inventor: Gregory Smith, filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1365US0)

[0010] VIRTUAL CONTENT REPOSITORY BROWSER, U.S. Application No. [[_____]] 10/618,379, Inventors: Jalpesh Patadia, et al., filed on [[_____]] July 11, 2003. (Attorney's Docket No. BEAS-1362US0)

Please replace paragraph [0030] with the following amended paragraph:

[0030] A virtual or federated content repository (hereinafter referred to as "VCR") **100** is a logical representation of one or more individual content repositories **108** such that they appear and behave as a single content repository from [[an application program's]] standpoint of an application program **110**. This is accomplished in part by use of an API (application program interface) **104** and an SPI (service provider interface) **102**. An API **104** describes how an application program, library or process **110** can

interface with some program logic or functionality. By way of a non-limiting illustration, a process can include a thread, a server, a servlet, a portlet, a distributed object, a web browser, or a lightweight process. An SPI 102 describes how a service provider (e.g., a content repository 108) can be integrated into a system of some kind. [[SPI's]] SPIs are typically specified as a collection of classes/interfaces, data structures and functions that work together to [[provided]] provide a programmatic means through which a service can be accessed and utilized. By way of a non-limiting example, APIs and SPIs can be specified in an object-oriented programming language, such as Java™ (available from Sun Microsystems, Inc. of Mountain View, California) and C# (available from Microsoft Corp. of Redmond, Washington). The API and SPI can be exposed in a number of ways, including but not limited to static libraries, dynamic link libraries, distributed objects, servers, class/interface instances, etc.

Please replace paragraph [0031] with the following amended paragraph:

[0031] In one embodiment, the API 104 presents a unified view of all repositories 108 to application programs 110 and enables them to navigate, perform CRUD (create, read, update, and delete) operations, and search across multiple content repositories 108 as though they were a single repository, or VCR 100. Content repositories 108 that implement the SPI 102 can "plug into" the VCR 100. The SPI includes a set of interfaces and services that repositories can implement and extend including schema management, hierarchy operations and CRUD operations. The API and SPI share a content model 106 that represents the combined content of all repositories 108 as a hierarchical namespace of nodes (or hierarchy). Given a node *N*, nodes that are hierarchically inferior to *N* are referred to as children of *N* whereas nodes that are hierarchically superior to *N* are referred to as parents of *N*. The top-most level of the hierarchy is called the federated root. There is no limit to the depth of the hierarchy.

Please replace paragraph [0032] with the following amended paragraph:

[0032] In one embodiment, content repositories 108 can be children of the federated root. Each content repository can have child nodes. Nodes can represent hierarchy information or content. Hierarchy nodes serve as a container for other nodes in the hierarchy akin to a file subdirectory in a hierarchical file system. Content nodes can have properties. In one embodiment, a property associates a name with a value of some kind. By way of a non-limiting illustration, a value can be a text string, a number, an image, an audio/visual presentation, binary data, etc. Either type of node can have a schema associated with it. A schema describes the data type of one or more of a node's properties.

Please replace paragraph [0036] with the following amended paragraph:

[0036] **Figure 3** is an illustration of objects used in connecting a repository to a VCR in one embodiment of the invention. In one embodiment, objects implementing API interface *RepositoryManager* **302** can serve as [[an]] a representation of a VCR from [[an application program's]] the standpoint of an application program **300**. A *RepositoryManager connect()* method attempts to connect all available repositories with a current user's credentials to the VCR. By way of a non-limiting example, credentials in one embodiment can be based on the Java™ Authentication and Authorization Service (available from Sun Microsystems, Inc.). Those of skill in the art will recognize that many authorization schemes are possible without departing from the scope and spirit of the present embodiment. Each available content repository **312-316** is represented by an SPI *Repository* object **306-310**. The *RepositoryManager* object invokes a *connect()* method on a set of *Repository* objects. In one embodiment, a *RepositorySession* object (not shown) can be instantiated for each content repository to which a connection is attempted. In one embodiment, the *RepositoryManager connect()* method can return an array of the *RepositiorySessions* to the application program, one for each repository for which a connection was attempted. Any error in the connection procedure can be described by the *RepositorySession* object's state. In another embodiment, the *RepositoryManager connect()* method can connect to a specific repository using a current user's credentials and a given repository name. In one embodiment, the name of a repository can be a URI (uniform resource identifier).

Please replace paragraph [0048] with the following amended paragraph:

[0048] **Figure 8** is an illustration of a content editor **800** in one embodiment of the invention. Navigation pane **802** is in "content" mode **812** such that it selectively filters out nodes that define only schemas. Content node **806** ("Laptop") has been selected. Node **806** is a child of hierarchy node "Products", which itself is a child of repository "BEA Repository". Selection of node **806** causes a corresponding content node editor to be rendered in editor window **804**. The editor displays the current values for the selected node. The content type **814** indicates that the schema for this node is named "product". In this example, the node has five properties: "Style", "Description", "Color", "SKU" and "Image". A user is allowed to change the value associated with these properties and update the VCR (via the update button **808**), or remove the node from the VCR (via the remove button **810**).

Please replace paragraph [0049] with the following amended paragraph:

[0049] **Figure 9** is an illustration of a schema editor **900** in one embodiment of the invention. Navigation pane **902** is in "type" mode **910** such that it only displays nodes that have schemas but no content. Schema node **906** ("product") has been selected. Node **906** is a child of repository "BEA

Repository". Selection of node **906** causes a corresponding schema editor to be rendered in editor window **904**. The editor displays the current schema for the selected node (e.g., derived from *ObjectClass, PropertyDefinition, PropertyChoice* objects). In this example, the node has five property definitions: "Style", "Description", "Color", "SKU" and "Image". For each property, the editor displays an indication of whether it is the primary property, its data type, its default value, and whether it is required. A property can be removed from a schema by selecting the property's delete button **912**. A property can be added by selecting the "add property" button **908**. A property's attributes can be changed by selecting its name **914** in the editor window or the navigation pane **906** (see **Figure 10**).

Please replace paragraph [0050] with the following amended paragraph:

[0050] **Figure 10** is an illustration of a property editor **1000** in one embodiment of the invention. The schema named "product" is being edited. Schema properties definitions are listed beneath their schema name in the navigation pane **1002**. Schema property **1008** ("color") has been selected. The editor window **1004** displays the property's current attributes. The name of the attribute (e.g., "color"), whether the attribute is required or not, whether it is read-only, whether it is the primary property, its data type, default value(s), and whether the property is single/multiple restricted/unrestricted can be modified. Changes to the a property's attributes can be saved by selecting the update button **1006**.

Please replace paragraph [0055] with the following amended paragraph:

[0055] A method of operating on a virtual content repository (VCR) that represents a number of content repositories having different types of content includes creating a hierarchy of hierarchy nodes in the VCR, indicating a location of the hierarchy node in the hierarchy by an identifier, relating each hierarchy node to a type of content and associating each hierarchy node with a first schema. A content node is created for each of the content repositories, and each hierarchy node is associated with at least one content node. Schemas are stored in one of the content repositories. Another method includes selecting a node from a hierarchy of both hierarchy nodes and content nodes in the VCR and performing an operation on the node, such as deleting the node, changing the location of the node in the VCR, reading the schema associated with the node and updating the schema associated with the node.